PCRemote

Project documentation

Kamil Janowski

PCRemote® All rights reserved © 2012

PCRemote

Documentation

# Project description

The whole project consists of 3 applications: server written with C++ and Qt Framework that will run on Windows operating system and two clients, one for Android and one for Windows Phone.

Clients connect with the server over the network via TCP (for general data transfer purposes) and UDP (for mouse movement requests, since UDP doesn’t perform any data correction checks, therefore doesn’t generate any delays) and will take the control over the cursor and the keyboard.

# Functional requirements

## Functional requirements for clients

1. It is possible to use mobile device like a touchpad to control cursor and scrolls
2. It is possible to use mobile device to control the keyboard (all keys that can be found on English keyboard)
3. It is possible to add and store data about a new server
4. It is possible to connect to the server
5. It is possible to choose sensitivity of a cursor
6. It is possible to define the time of a long click (how long the control needs to be pressed to call it a long click)
7. It is possible to define sensitivity of a scroll

## Functional requirements for the server

1. Server notifies of each new client connection
2. Server notifies of each client disconnection
3. Server processes all client requests concerning cursor movement or key press
4. Has a tray icon
5. Is able to provide information required to connect to it
6. It is possible to change default port for listening
7. It has an option for closing the application from tray icon menu

# Non-functional requirements

## Non-functional requirements for server

1. Windows Vista / 7 / 8 operating system
2. Wireless internet connection

## Non-functional requirements for clients

1. Either Windows Phone 7.1 or Android (at least 2.3 version)
2. Access to wireless internet connection

# Why these technologies

You might wonder, why among non-functional requirements for server you can find Windows, but not Linux even though it’s supposed to be written with Qt Framework which is multiplatform framework for C++. Well, the problem is, that even though Qt Framework makes it much easier to write applications with graphical user interface, it still doesn’t provide all the features required by the project, such as simulation of mouse or keyboard button click. What’s more, Linux native libraries do not provide this functionality either while WinAPI does.

Then why is it written in C++ at all? It’s because C++ is very efficient language and efficiency for background applications is very important. You certainly don’t want any background application that doesn’t do anything for most of the time (unless you ask it to) to use up 20 MB of memory. What’s more applications written in C++ are totally stand-alone. Many potential users of the application have very little experience with usage of computers and satisfying a requirement of installing for instance JRE before application can run, might be unachievable task for them (especially since nobody likes reading installation instructions).

As for clients, Android was chosen because it’s currently the most popular mobile operating system, and even though version 2.3 is already pretty old, it’s still the most popular version of Android.

Windows Phone recently started gaining on popularity but yet there are very few native applications for this operating system, what means, there are very few competitive applications.

Why not IOS then? It’s also very popular after all.

Well, there are two reasons for that:

1. In order to code applications for IOS, you need Mac, which I don’t have
2. I don’t know anyone, that could lend me an iPhone for testing
3. I don’t like Apple

# User stories

## PCRemote Server

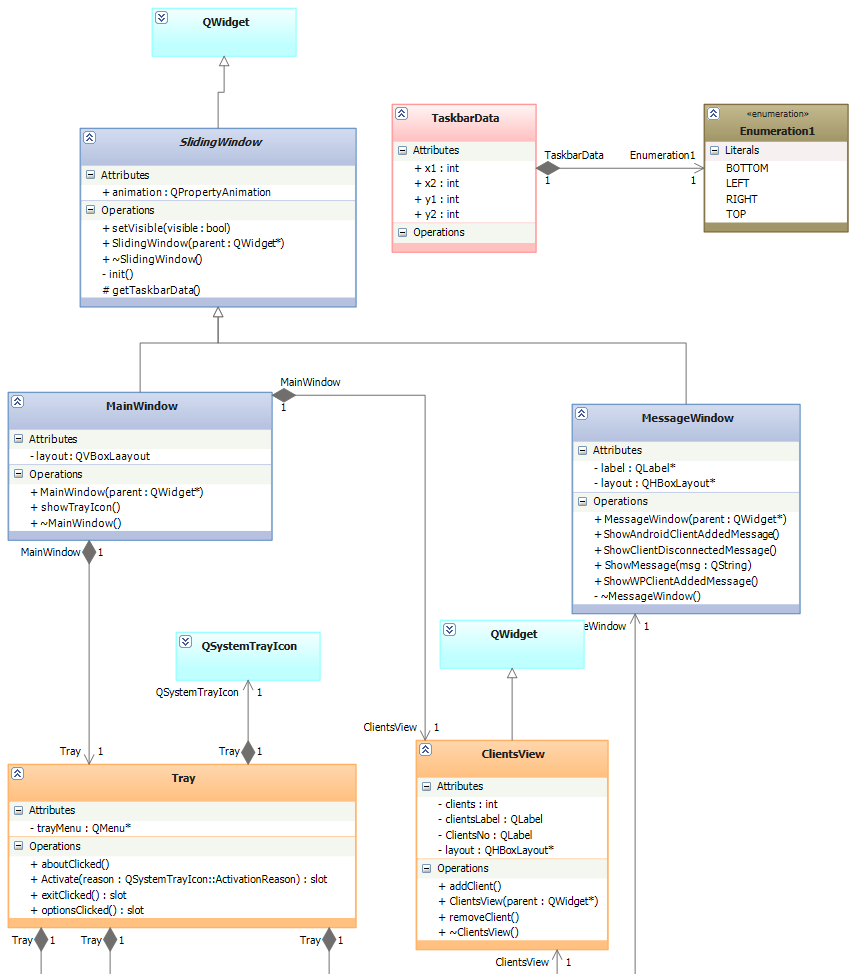
1. As a user of a server, I want to access all its features through a tray icon.
2. As a user I want the server to support at least all basic functions of controlling my computer, such as full control over the cursor, just the way I control it with touchpad, control over keyboard and basic media keys (play, previous, next).
3. As a user of a server I want to be able to set it up, so that it refused connections from certain clients.
4. As a user I want to server to be able to provide me instructions of how I should use it and its clients.
5. As a user I want to be able to choose whether my server can be detected automatically by a client.

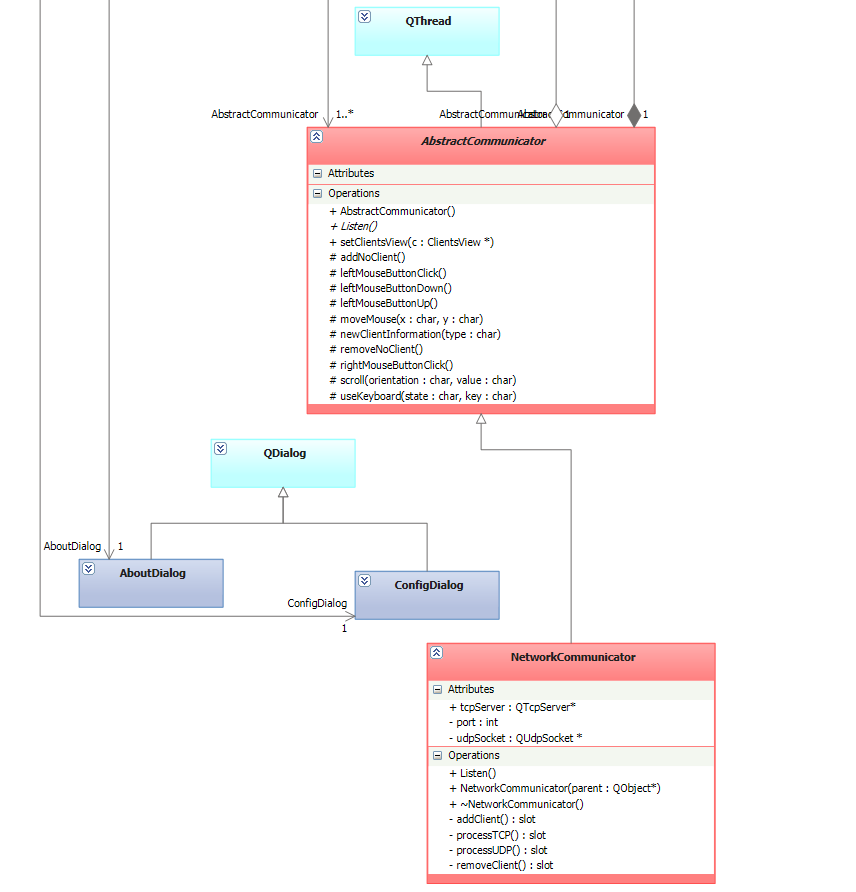
## Clients

1. As a user I want the application to automatically detect all the servers working in the network, my device is connected to.
2. As a user I want to be able to add information about a server manually, since some of the servers might be set to “undetectable” mode.
3. As a user I want to be able to control my cursor as if my device was a real touchpad.
4. As a user I want to be able to control my keyboard via device with client installed.
5. As a user I want to be able to control my media buttons: play, next, previous, volume up, volume down.
6. As a user I want to be able to choose sensitivity of touchpad.
7. As a user I want to be able to choose sensitivity of scrolls.

# Diagrams

## PCRemote Server

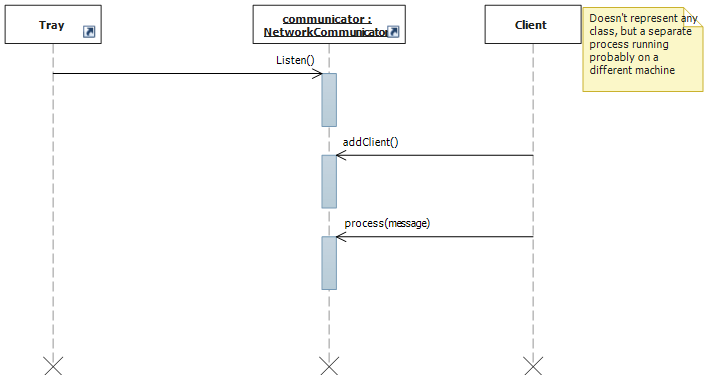




Main part of the whole application is Tray class. Even though it’s still just a part of MainWindow, it is visible all the time, while MainWindow is visible only after Tray is activated (tray icon is clicked by a user). Tray contains a set of AbstractCommunicators, although in this example we’ll use only NetworkCommunicator (inherits AbstractCommunicator) that allows to communicate with the clients via TCP (general events) and UDP (mouse movement event). This is also Tray that calls the Listen method in each AbstractCommunicator. Then Tray also contains the menu (visible after clicking tray icon with right mouse button) for managing the whole application.



As a matter of fact, ClientsView is nothing more than just a Widget consisting of two labels. One that says “Number of clients” and the other one that actually represents the number.



## PCRemote Client



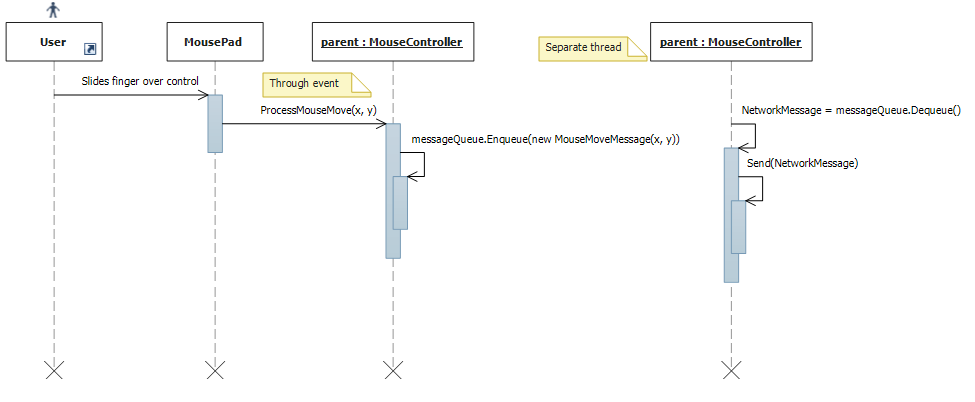


Data contained and generated by ServersStorage are common for all classes across the application, therefore it’s much easier to make the entire class static, rather than pass a reference to it to all the objects used in the application.



Mouse controller is a UserControl that contains two other user controls: MousePad and MouseScroll.

Each time MousePad or MouseScroll recognize a gesture, they raise an event that is captured later on by MouseController. Then depending on the event MouseController generates a proper NetworkMessage and puts it into a messageQueue. Within MouseController there is a separate thread working in the background, that continuously reads the messageQueue and sends data to the server.





# Data dictionary

1. Client – any application that is able to communicate with the PCRemote Server
2. Mobile device – device running either Windows Phone (at least 7.1 version) or Android (at least 2.3 version) operating system with client application installed
3. MessageQueue – Queue (First-In-First-Out) containing set of NetworkMessage objects.